

F²MC-8FX FAMILY

8-BIT MICROCONTROLLER

MB95F430 SERIES

16-bit PPG Timer

APPLICATION NOTE



Revision History

Date	Author	Change of Records
2010-03-16	Folix	V1.0, First draft

This manual contains 28 pages.

1. The products described in this manual and the specifications thereof may be changed without prior notice. To obtain up-to-date information and/or specifications, contact your Fujitsu sales representative or Fujitsu authorized dealer.
2. Fujitsu will not be liable for infringement of copyright, industrial property right, or other rights of a third party caused by the use of information or drawings described in this manual.
3. The contents of this manual may not be transferred or copied without the express permission of Fujitsu.
4. The products contained in this manual are not intended for use with equipment which require extremely high reliability such as aerospace equipments, undersea repeaters, nuclear control systems or medical equipments for life support.
5. Some of the products described in this manual may be strategic materials (or special technology) as defined by the Foreign Exchange and Foreign Trade Control Law. In such cases, the products or portions thereof must not be exported without permission as defined under the law.

Table of Contents

Revision History	2
Table of Contents.....	3
1 Introduction	4
2 PPG Overview	5
2.1 Background	5
2.2 PPG Formats.....	5
3 16bit PPG Timer Description	6
3.1 Block Diagram of 16bit PPG Timer.....	7
3.2 Pins of 16bit PPG Timer	7
3.3 Registers List of 16bit PPG Timer	7
3.4 16bit PPG Down Counter Registers	8
3.5 16bit PPG Cycle Setting Buffer Registers	9
3.6 16bit PPG Duty Setting Buffer Registers	10
3.7 16bit PPG Status Control Registers	11
3.8 16bit PPG trigger source control Register	13
3.9 PPG Macro general setting procedure	14
4 Interrupts of 16bit PPG Timer	15
4.1 Interrupt Control Bits and Interrupt Sources of 16-bit PPG Timer.....	15
4.2 Register and Vector Table Addresses Related to Interrupts of 16bit PPG Timer.....	15
5 PPG Trigger	16
6 PPG Driver.....	19
6.1 Peripheral Usage.....	19
6.2 Driver Code	19
6.2.1 General Definition	19
6.2.2 PPG Routine	20
7 Typical Application.....	21
7.1 HW Design	21
7.2 Sample Code.....	21
8 More Information.....	23
9 Appendix.....	24
10 Sample Code.....	25

1 Introduction

This application note introduces how to use the PPG function on MB95F430 series.

Chapter 2 introduces the background and PPG formats.

Chapter 3 introduces PPG block, PPG registers, and the setting procedure.

Chapter 4 introduces PPG interrupt.

Chapter 5 introduces PPG drivers.

Chapter 6 introduces PPG application demo.

2 PPG Overview

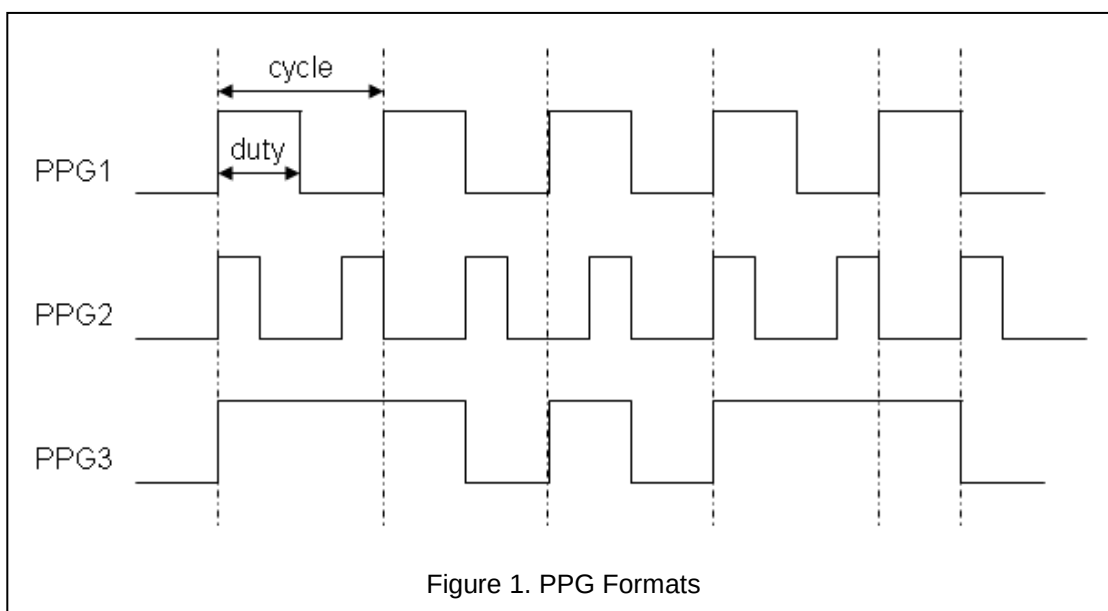
Chapter 2 is an overview of PPG.

2.1 Background

It is generally used in communication, electronics, automation and so on. In communication, the duty or frequency can be used to identify the protocol. In electronics, it can be used to design switching power supply. In automation, it can be used to control motors.

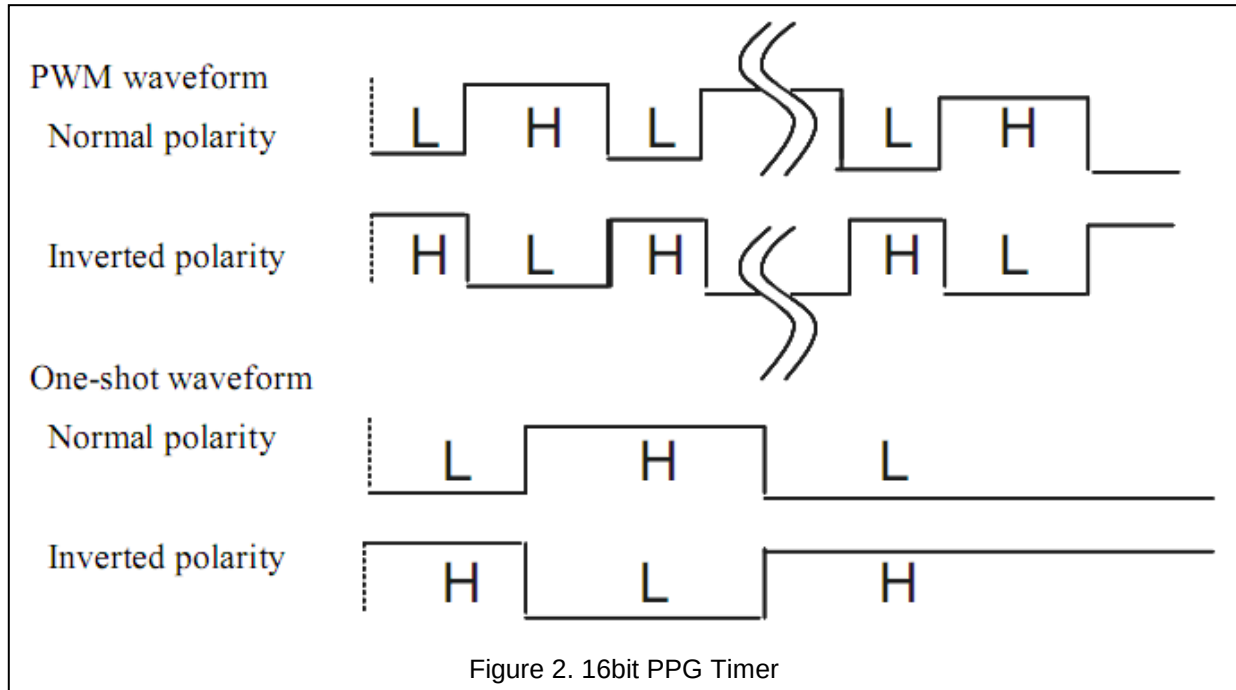
2.2 PPG Formats

PPG format consists of frequency and duty. PPG's waveform is shown as below.

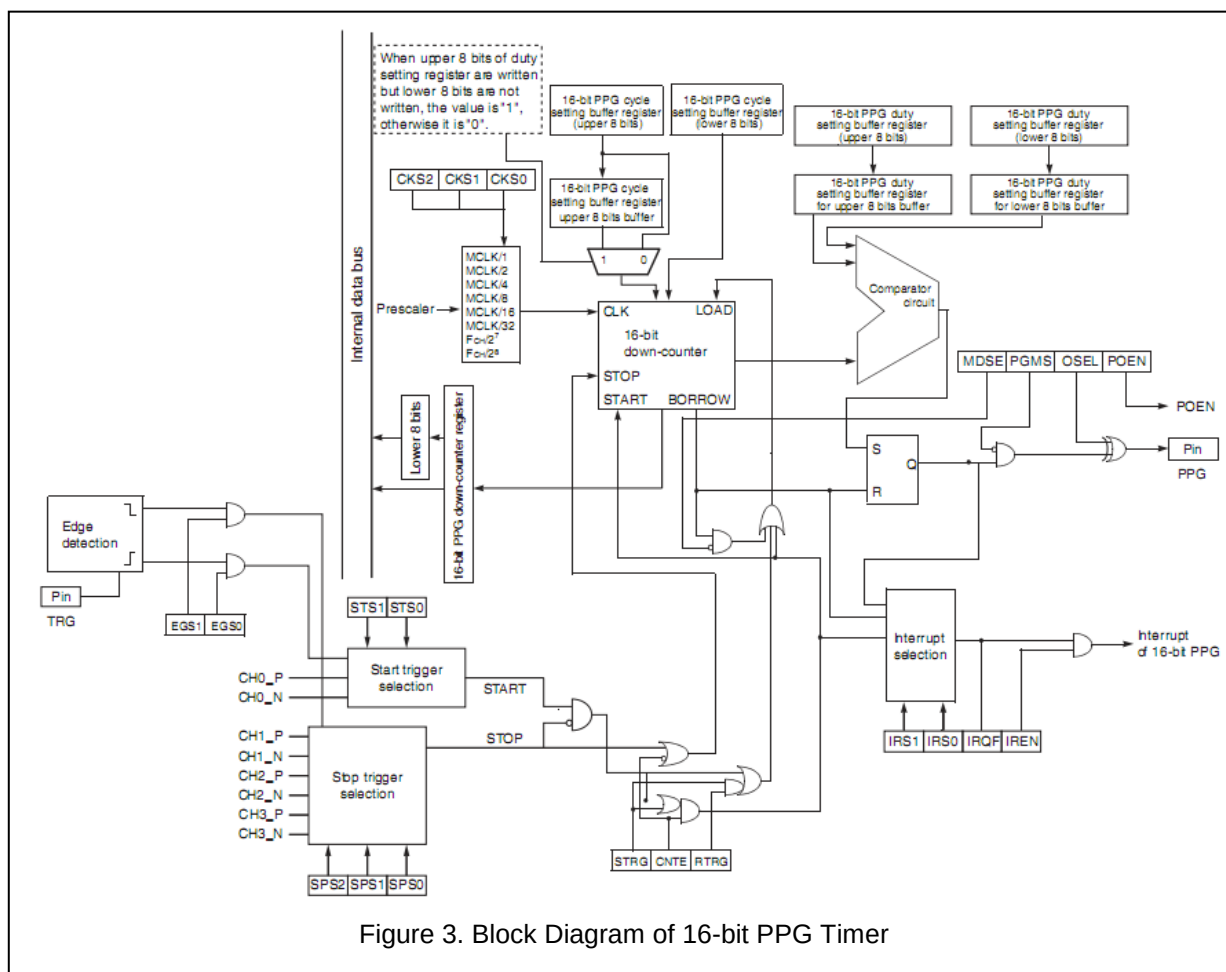


3 16bit PPG Timer Description

16-bit PPG timer can output the waveform of PWM and one shot. The output waveform can be reversed by setting the register (Normal polarity ↔ Inverted polarity).



3.1 Block Diagram of 16bit PPG Timer



3.2 Pins of 16bit PPG Timer

The pins related to the 16-bit PPG timer namely are: PPG0 pin, TRG0 pin. PPG0 is output pin. The PPG waveform can be outputted by using the 16-bit PPG status control register to enable output (PCNTL0: POEN=1). TRG0 is used to start 16-bit PPG timer by hardware trigger.

3.3 Registers List of 16bit PPG Timer

Register	Description
PDCRH0	16-bit PPG down counter register (upper)
PDCRL0	16-bit PPG down counter register (lower)
PCSRH0	16-bit PPG cycle setting buffer register (upper)
PCSRL0	16-bit PPG cycle setting buffer register (lower)
PDUTH0	16-bit PPG duty setting buffer register (upper)
PDUTL0	16-bit PPG duty setting buffer register (lower)
PCNTH0	16-bit PPG status control register (upper)
PCNTL0	16-bit PPG status control register (lower)
PTGS0	16-bit PPG trigger source control register

3.4 16bit PPG Down Counter Registers

The 16-bit PPG down counter registers (upper, lower) (PDCRH0, PDCRL0) form a 16-bit register which is used to read the count value from the 16-bit PPG down-counter. These registers form a 16-bit register which is used to read the count value from the 16-bit down-counter.

The initial values of the register are all "0".

Always use one of the following procedures to read from this register.

- Use "MOVW" instruction (use a 16-bit access instruction to read the PDCRH0 register address)

- Use "MOV" instruction, and read PDCRH0 first and then PDCRL0 (reading PDCRH0 automatically copies the lower 8 bits of the down-counter to PDCRL0).

These registers are read-only and writing has no effect on the operation.

16-bit PPG down counter register (upper) PDCRH0									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0FAA _H PDCRH0	DC15	DC14	DC13	DC12	DC11	DC10	DC09	DC08	00000000 _B
	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	
16-bit PPG down counter register (lower) PDCRL0									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0FAB _H PDCRL0	DC07	DC06	DC05	DC04	DC03	DC02	DC01	DC00	00000000 _B
	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	R/WX	

Figure 4. 16bit PPG Down Counter Register

Figure 4. 16bit PPG Down Counter Register

3.5 16bit PPG Cycle Setting Buffer Registers

The 16-bit PPG cycle setting buffer registers are used to set the cycle for the output pulses generated by the PPG.

These registers form a 16-bit register which sets the period for the output pulses generated by the PPG. The values set in these registers are loaded to the down-counter.

When writing to these registers, always use one of the following procedures.

- Use "MOVW" instruction (use a 16-bit access instruction to write to the PCSRH0 register address)

- Use "MOV" instruction, and write to PCSRH0 first and then PCSRL0.

If a down-counter load occurs after writing data to PCSRH0 (but before writing data to PCSRL0), the previous valid PCSRH0/PCSRL0 value will be loaded to the down-counter. If the PCSRH0/PCSRL0 value is modified during counting, the modified value will become effective from the next load of the down-counter.

- Do not set PCSRH0 and PCSRL0 to "00H", or PCSRH0 to "01H" and PCSRL0 to "01H".

16-bit PPG cycle setting buffer register (upper) PCSRH0									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0FAC _H PCSRH0	CS15	CS14	CS13	CS12	CS11	CS10	CS09	CS08	11111111 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
16-bit PPG cycle setting buffer register (lower) PCSRL0									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0FAD _H PCSRL0	CS07	CS06	CS05	CS04	CS03	CS02	CS01	CS00	11111111 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Figure 5. 16bit PPG Cycle Register

3.6 16bit PPG Duty Setting Buffer Registers

The 16-bit PPG duty setting buffer registers control the duty ratio for the output pulses generated by the PPG.

These registers form a 16-bit register which controls the duty ratio for the output pulses generated by the PPG. Transfer of the data from the 16-bit PPG duty setting buffer registers to the duty setting registers is performed at the same timing as the down-counter read.

When writing to these registers, always use one of the following procedures.

- Use "MOVW" instruction (use a 16-bit access instruction to write to the PDUTH0 register address).

- Use "MOV" instruction, and write to PDUTH0 first and then PDUTL0.

If a down-counter load occurs after writing data to PDUTH0 (but before writing data to PDUTL0), the value of the 16-bit PPG duty setting buffer registers is not transferred to the duty setting registers.

The relation between the value of the 16-bit PPG duty setting registers and output pulse is as follows:

- When the same value is set in both the 16-bit PPG cycle setting buffer registers and duty setting registers, the "H" level will always be outputted if normal polarity is set, or the "L" level will always be outputted if inverted polarity is set.

- When the duty setting registers are set to "00B", the "L" level will always be outputted if normal polarity is set, or the "H" level will always be outputted if inverted polarity is set.

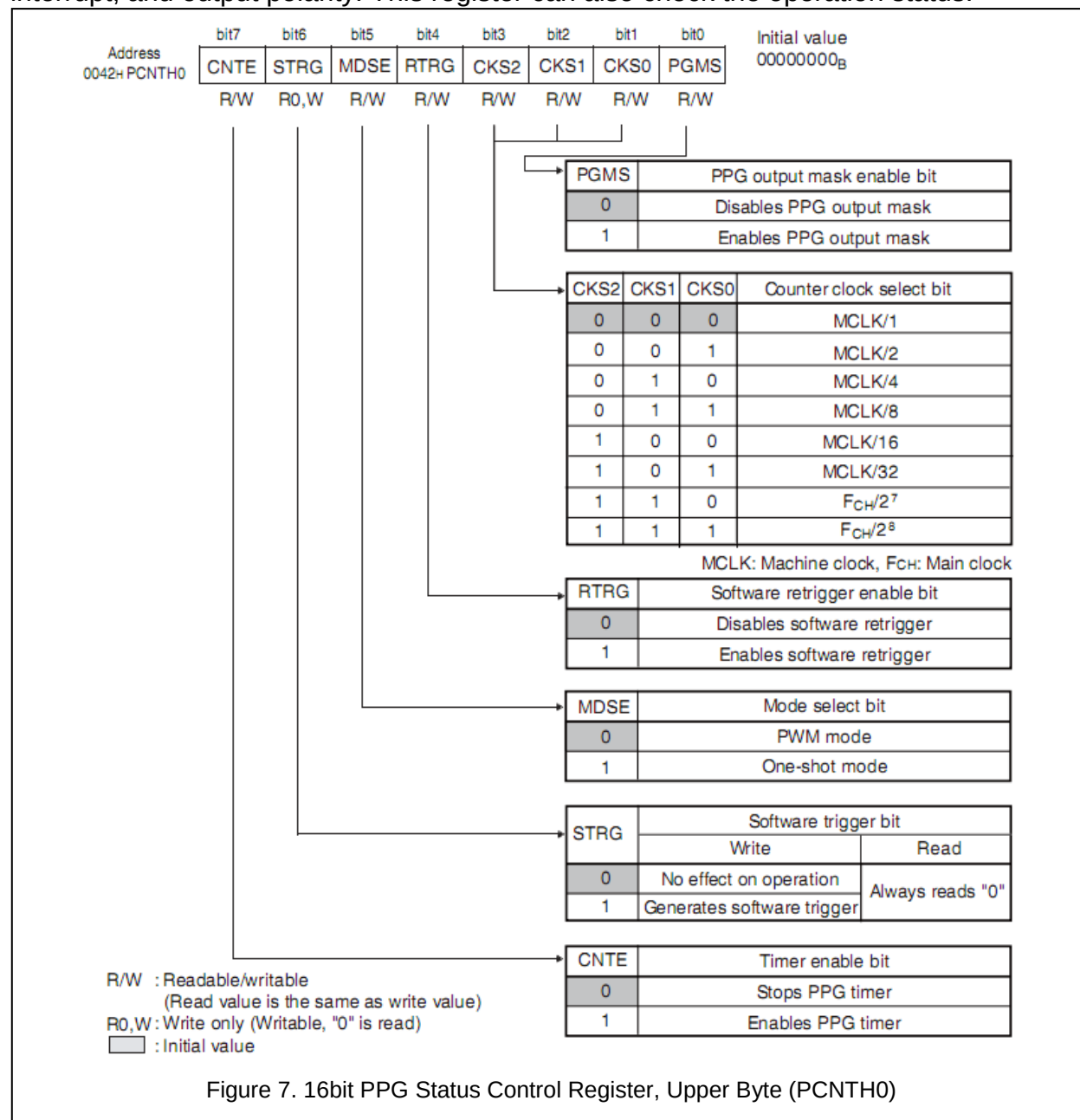
- When the value set in the duty setting registers is greater than the value in the 16-bit PPG cycle setting buffer registers, the "L" level will always be outputted if normal polarity is set, and the "H" level will always be outputted if inverted polarity is set.

16-bit PPG duty setting buffer register (upper) PDUTH0									
Address	bit15	bit14	bit13	bit12	bit11	bit10	bit9	bit8	Initial value
0FAE _H PDUTH0	DU15	DU14	DU13	DU12	DU11	DU10	DU09	DU08	11111111 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
16-bit PPG duty setting buffer register (lower) PDUTL0									
Address	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Initial value
0FAF _H PDUTL0	DU07	DU06	DU05	DU04	DU03	DU02	DU01	DU00	11111111 _B
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

Figure 6. 16bit PPG Duty Register

3.7 16-bit PPG Status Control Registers

The 16-bit PPG status control register is used to enable and disable the 16-bit PPG timer and also to set the operating status for the software trigger, retrigger control interrupt, and output polarity. This register can also check the operation status.



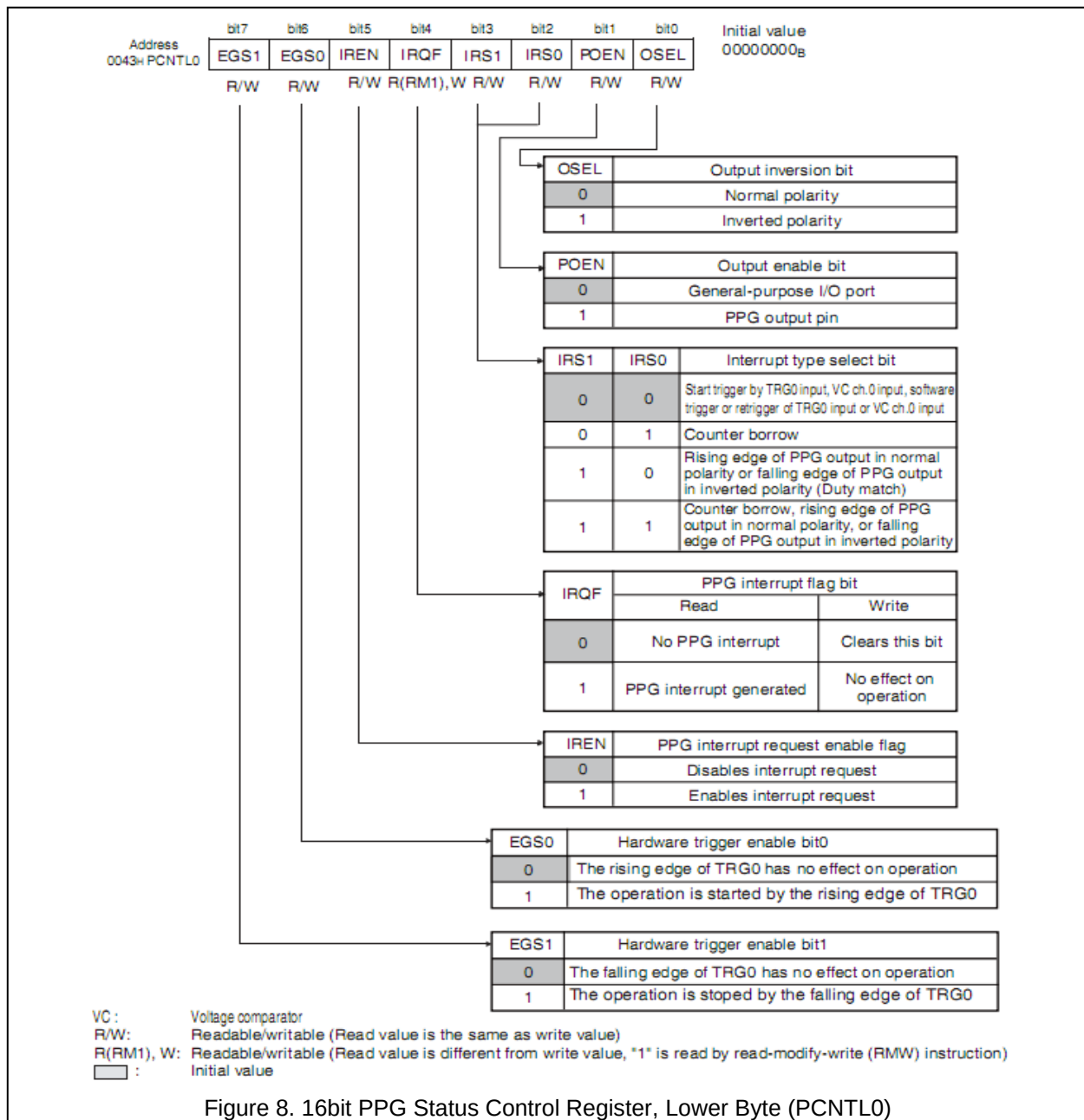
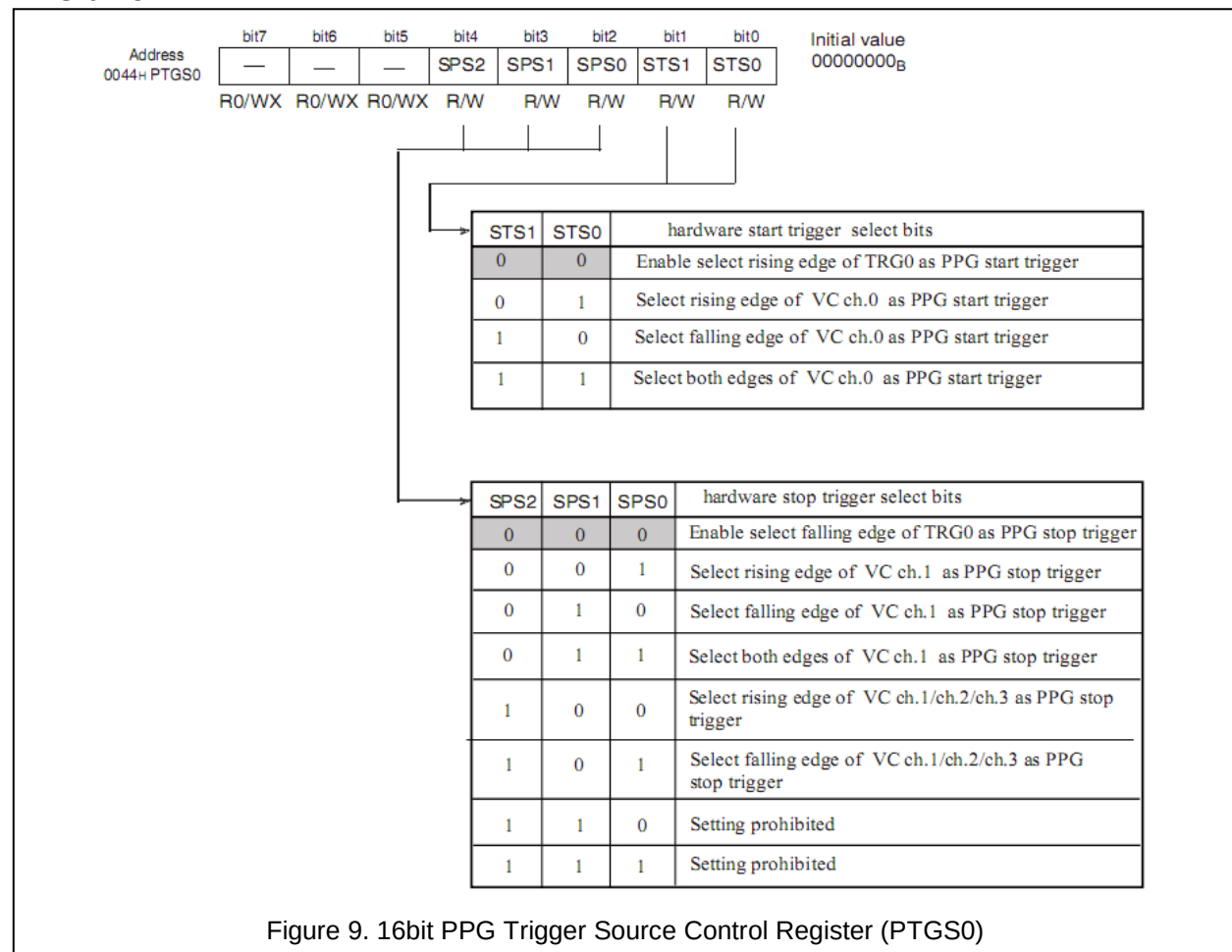


Figure 8. 16bit PPG Status Control Register, Lower Byte (PCNTL0)

3.8 16-bit PPG Trigger Source Control Register

The 16-bit PPG trigger source control register controls the trigger source of the 16-bit PPG timer.



3.9 PPG Macro General Setting Procedure

- Initial setup

- 1) Set the interrupt level. (ILR3, ILR4)
- 2) Enable the hardware trigger and interrupts, select the interrupt type, and enable output. (PCNTL0)
- 3) Select the count clock and the mode, and enable timer operation. (PCNTH0)
- 4) Set the cycle. (PCSRH0, PCSRL0)
- 5) Set the duty. (PDUTH0, PDUTL0)
- 6) Start the PPG by the software trigger. (PCNTH0: STRG = 1)

- Interrupt processing

- 1) Process any interrupt.
- 2) Clear the interrupt request flag. (PCNTL0: IRQF)

4 Interrupts of 16bit PPG Timer

This chapter introduces PPG interrupt

The 16-bit PPG timer can generate interrupt requests in the following cases:

- When a trigger or counter borrow occurs
- When a rising edge of PPG is generated in normal polarity
- When a falling edge of PPG is generated in inverted polarity

The interrupt operation is controlled by IRS1 (bit3) and IRS0 (bit2) in the PCNTL register.

4.1 Interrupt Control Bits and Interrupt Sources of 16-bit PPG Timer

Item	Description
Interrupt flag bit	PCNTL0:IRQF
Interrupt request enable bit	PCNTL0:IREN
Interrupt type select bits	PCNTL0:IRS1, IRS0
Interrupt sources	PCNTL0:IRS1, IRS0=00 _B Hardware trigger by TRG Pin input of 16-bit down-counter, software trigger and retrigger
	PCNTL0:IRS1, IRS0=01 _B Counter borrow of 16-bit down-counter
	PCNTL0:IRS1, IRS0=10 _B Rising edge of PPG1 output in normal polarity, or falling edge of PPG1 output in inverted polarity
	PCNTL0:IRS1, IRS0=11 _B Counter borrow of 16-bit down-counter, rising edge of PPG1 output in normal polarity, or falling edge of PPG1 output in inverted polarity

4.2 Register and Vector Table Addresses Related to Interrupts of 16bit PPG Timer

Interrupt source	Interrupt request no.	Interrupt level setting register		Vector table address	
		Register	Setting bit	Upper	Lower
16-bit PPG timer ch. 1	IRQ17	ILR4	L17	FFD8 _H	FFD9 _H

5 PPG Trigger

This chapter introduces PPG trigger.

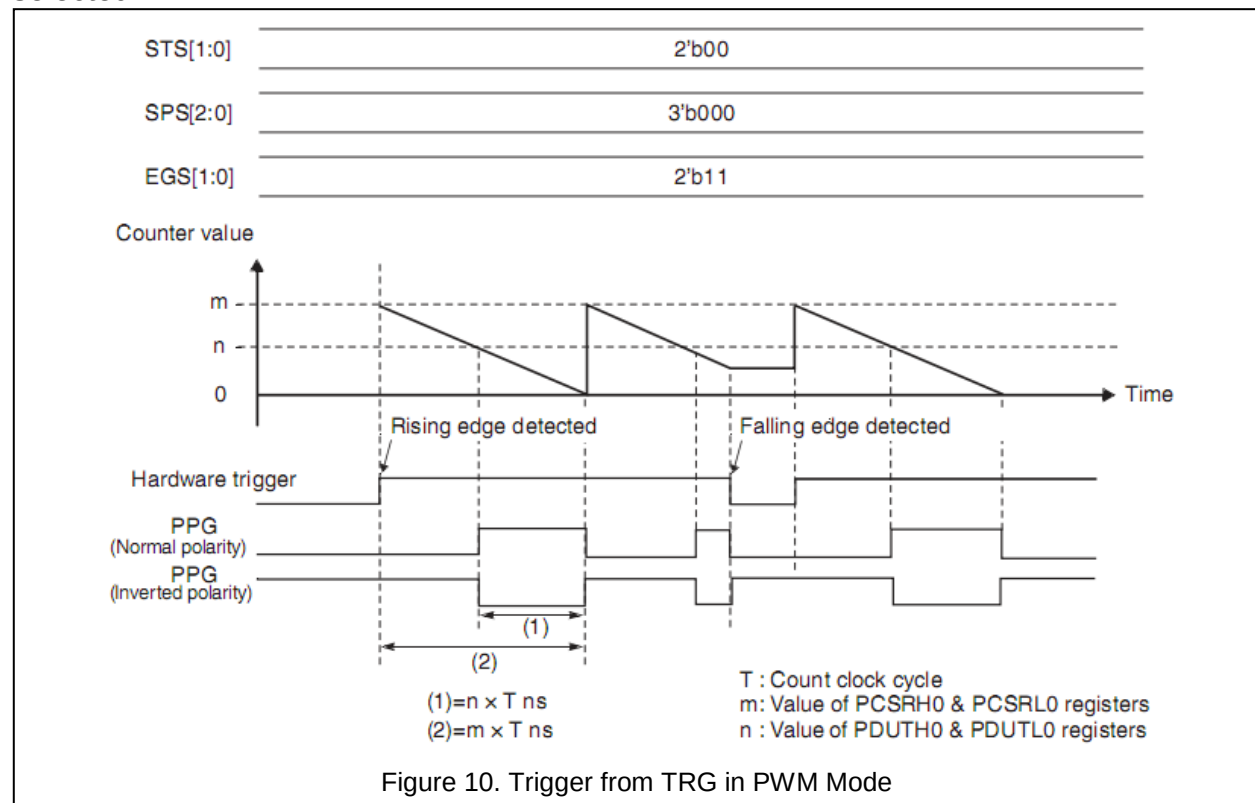
PPG is activated by TRG input or internal voltage comparator (VC) input.

Condition 1: PPG start triggering and stop triggering do not occur simultaneously.

(1) TRG

When STS1 and STS0 are set to "00B", SPS2, SPS1 and SPS0 are set to "000B", EGS1 and EGS0 are set to "11B", and the hardware trigger input from TRG is used, the 16-bit PPG timer starts operating at a rising edge and stops upon the detection of a falling edge. Moreover, the 16-bit PPG timer also starts operating at the following rising edge from the beginning.

The operation can be re-triggered by a valid TRG input hardware trigger regardless of the retrigger setting of the RTRG bit when the TRG input hardware trigger has been selected.

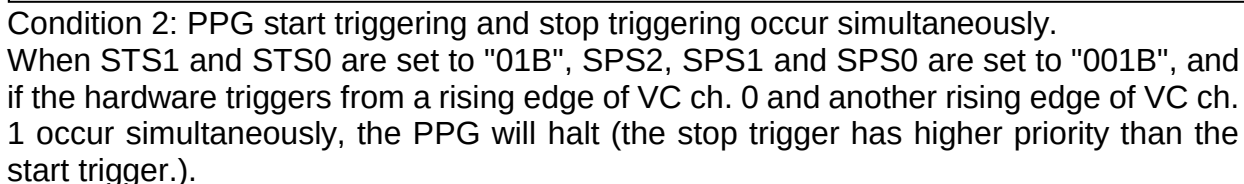


(2) Voltage comparator (VC)

When STS1 and STS0 are set to "01B", SPS2, SPS1 and SPS0 are set to "101B", and the hardware trigger input from the VC ch. 0 is used, the 16-bit PPG timer starts operating at a rising edge of VC ch. 0 and stops upon the detection of a falling edge of VC ch.1/2/3.

Moreover, the 16-bit PPG timer also starts operating at the following rising edge of VC ch. 0 from the beginning.

The operation can be re-triggered by a valid rising edge of VC ch. 0 input hardware trigger regardless of the retrigger setting of the RTRG bit when the VC ch. 0 input hardware trigger has been selected.



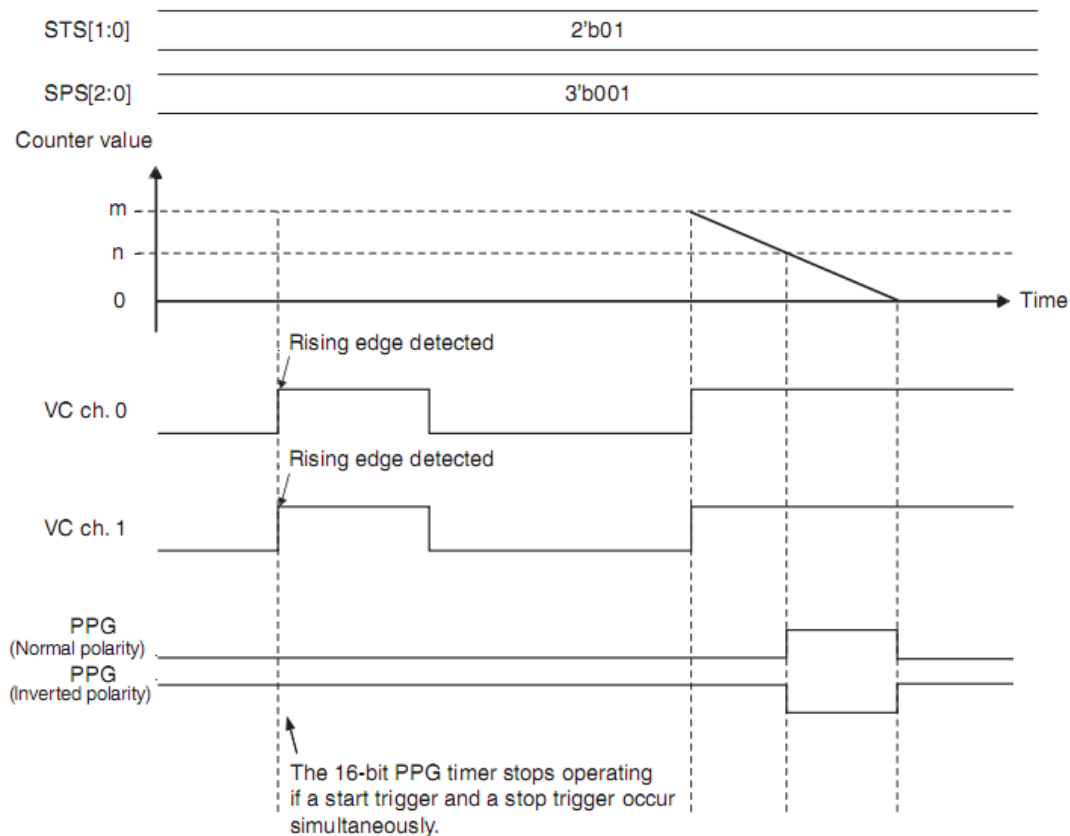


Figure 12. Trigger Start and Stop Occur Simultaneously in PWM Mode

6 PPG Driver

The chapter introduces PPG driver.

6.1 Peripheral Usage

The MCU pins are used as below:
PPG0 is used as PPG wave output;
TRG0 is used as PPG trigger.

6.2 Driver Code

6.2.1 General Definition

```
typedef unsigned char  BOOLEAN;
typedef unsigned char  INT8U;           /* Unsigned  8 bit quantity */
typedef signed   char  INT8S;           /* Signed    8 bit quantity */
typedef unsigned int   INT16U;          /* Unsigned 16 bit quantity */
typedef signed   int   INT16S;          /* Signed   16 bit quantity */
typedef unsigned long  INT32U;          /* Unsigned 32 bit quantity */
typedef signed   long  INT32S;          /* Signed   32 bit quantity */

#define BOOL          BOOLEAN
#define BYTE          INT8U
#define UBYTE         INT8U
#define WORD          INT16U
#define UWORD         INT16U
#define LONG          INT32S
#define ULONG         INT32U
#define UCHAR         INT8U
#define UINT          INT16U
#define DWORD         INT32U

#define TRUE          1
#define FALSE         0

#define BYTE_LO(w)    ((UBYTE)(w))
#define BYTE_HI(w)    ((UBYTE)((((UWORD)(w)>>8)&0xFF))
```

6.2.2 PPG Routine

void PPGOpen()

Return : none.
Parameters : none.
Description : open the PPG function.
Example : PPGOpen();

```
void PPGOpen ()
{
    DDR7_P73=1;
    PCNTH0_PGMS=0; //Open PPG
}
```

void PPGClose ()

Return : none.
Parameters : none.
Description : close the PPG function.
Example : PPGClose();

```
void PPGClose ()
{
    DDR7_P73=1;
    PCNTH0_PGMS=1; //Close PPG
}
```

void PPGSet(WORD usCycle,WORD usDuty)

Return : none.
Parameters : usCycle, is PPG cycle(us)
usDuty, is PPG duty(us)
Description : set the PPG function.
Example : PPGSet(40,10);

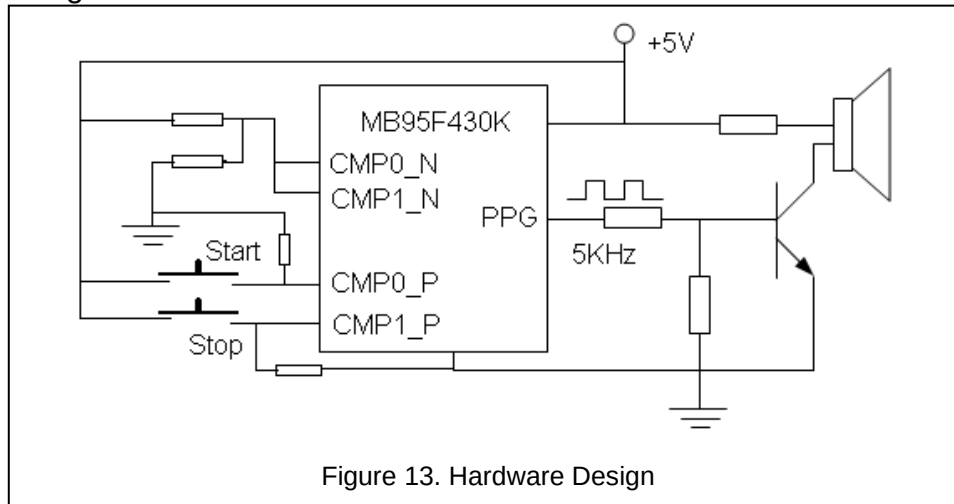
```
void PPGSet (WORD usCycle,WORD usDuty)
{
    DDR7_P73=1;
    //PPG0
    PCSRH0=BYTE_HI(usCycle); //Upper Byte
    PCSRL0=BYTE_LO(usCycle); //Lower Byte
    PDUTH0=BYTE_HI(usDuty); //Upper Byte
    PDUTL0=BYTE_LO(usDuty); //Lower Byte
    PCNTH0=0xF7; //Mask=1, 1.000us (MCLK/8), OneShot, Soft Trigger
    PCNTL0=0x0E; //OSEL=L, IREN=0
}
```

7 Typical Application

The chapter introduces the typical application of PPG.

7.1 HW Design

In this application, we will use the PPG to drive a buzzer. VC0 rising edge starts PPG, and VC1 rising edge stops PPG (PTGS0=0x05). The MCU is MB95F430K. The HW is designed as below.



7.2 Sample Code

```
void main(void)
Return      : none.
Parameters  : none;
Description : system main program.
Example     : main();
```

```

void main(void)
{
    __DI();
    __set_il(3);
    InitIrqLevels();

    WDTN = 0xA5; //Disable WDT
    WDTL = 0x96;

    WATR = 0xEE;
    SYCC = 0xF0; //Main Clock
    SYCC2 = 0xF4; //Main Clock
    SYSC = 0xBC; //BUZZ(P01)
    SYSC2 = 0x02; //PPG(P73), Disable I2C
    while(!STBC_MRDY);
    __EI();

    PTGS0 = 0x05; //VC0 rising edge start PPG, VC1 rising edge stop PPG
    PPGSet(200, 10);
    PPGOpen();
    PPGClose();
}

```

8 More Information

For more information on FUJITSU MB95200 products, please visit following website:

English Version <http://www.fujitsu.com/cn/fmc/en/services/mcu/mb95430/>

Simplified Chinese Version <http://www.fujitsu.com/cn/fmc/services/mcu/mb95430/>

9 Appendix

Figure 1. PPG Formats	5
Figure 2. 16bit PPG Timer	6
Figure 3. Block Diagram of 16-bit PPG Timer	7
Figure 4. 16bit PPG down counter register	8
Figure 5. 16bit PPG cycle register	9
Figure 6. 16bit PPG duty register	10
Figure 7. 16bit PPG Status Control Register, Upper Byte (PCNTH0).....	11
Figure 8. 16bit PPG Status Control Register, Lower Byte (PCNTL0).....	12
Figure 9. 16bit PPG trigger source control Register (PTGS0).....	13
Figure 10. Trigger from TRG in PWM Mode	16
Figure 11. Trigger from VC in PWM Mode.....	17
Figure 12. Triggers Start and Stop Occur Simultaneously in PWM Mode.....	18
Figure 13. Hardware design	21

10 Sample Code

main.c

```
/*-----*/
/* Title   : Sample main program for sensorless DC inverter control      */
/* Author   : Folix Li                                                    */
/* Date    : 26th Nov 2999                                                */
/*-----*/
#include "mb95430.h"
#include "TypeDef.h"

/*-----*/
/* PWM Control                                                            */
/*-----*/
void PPGOpen()
{
    DDR7_P73=1;
    PCNTH0_PGMS=0;
}

void PPGClose()
{
    DDR7_P73=1;
    PCNTH0_PGMS=1;
}

void PPGSet(WORD usCycle,WORD usDuty)
{
    DDR7_P73=1;
    //PPG0
    PCSRH0=BYTE_HI(usCycle);//Upper Byte
    PCSRL0=BYTE_LO(usCycle);//Lower Byte
    PDUTH0=BYTE_HI(usDuty);//Upper Byte
    PDUTL0=BYTE_LO(usDuty);//Lower Byte
    PCNTH0=0xF7;//Mask=1,1.000us(MCLK/8),OneShot,Soft Trigger
    PCNTL0=0x0E;//OSEL=L,IREN=0
}

void main(void)
{
    __DI();
    __set_il(3);
    InitIrqLevels();

    WDTN =0xA5;
    WDTL =0x96;
```

```

WATR =0xEE;
SYCC =0xF0;//Main Clock
SYCC2=0xF4;//Main Clock
SYSC  =0xBC;//BUZZ(P01)
SYSC2 =0x02;//PPG(P73),Disable I2C
while(!STBC_MRDY);
__EI();

PTGS0=0x05;//VC0 rising edge start PPG,VC1 rising edge stop PPG
PPGSet(200,10);
PPGOpen();
PPGClose();
}

```

VECTORS.C

```

/* THIS SAMPLE CODE IS PROVIDED AS IS AND IS SUBJECT TO ALTERATIONS.
FUJITSU */
/* MICROELECTRONICS ACCEPTS NO RESPONSIBILITY OR LIABILITY FOR ANY
ERRORS OR */
/*          ELIGIBILITY          FOR          ANY          PURPOSES.
*/
/*          (C) Fujitsu Microelectronics Europe GmbH          */
/*-----*/

```

VECTORS.C

```

- Interrupt level (priority) setting
- Interrupt vector definition
29.09.04  1.00  HWe  V30L29

```

```

-----*/

```

```

#include "mb95430.h"

```

```

/*-----*/
InitIrqLevels()

```

This function pre-sets all interrupt control registers. It can be used to set all interrupt priorities in static applications. If this file contains assignments to dedicated resources, verify that the appropriate controller is used.

NOTE: value 0xFF disables the interrupt and value 0 sets highest priority.

NOTE: For all resource interrupts exists 3 Interrupt level registers (ILRx).

Each register sets the level for 4 different resources (IRQx).

NOTE: This list is prepared for the 8FX-emulation chip MB95FV100 'Horn'.

Not all resources will be supported by all 8FX-devices

```

-----*/

```

```

void InitIrqLevels(void)
{

```

```

/* ILRx                                IRQs defined by ILRx */

    ILR0 = 0xFF;    // IRQ0:  external interrupt ch0 | ch4
                   // IRQ1:  external interrupt ch1 | ch5
                   // IRQ2:  external interrupt ch2 | ch6
                   // IRQ3:  external interrupt ch3 | ch7

    ILR1 = 0xFF;    // IRQ4:  UART/SIO ch0
                   // IRQ5:  8/16-bit timer ch0 (lower)
                   // IRQ6:  8/16-bit timer ch0 (upper)
                   // IRQ7:  Output Compare ch0

    ILR2 = 0xFF;    // IRQ8:  Output Compare ch1
                   // IRQ9:  none
                   // IRQ10: Voltage Compare ch0
                   // IRQ11: Voltage Compare ch1

    ILR3 = 0xFF;    // IRQ12: Voltage Compare ch2
                   // IRQ13: Voltage Compare ch3
                   // IRQ14: 16-bit free run timer
                   // IRQ15: 16-bit PPG0

    ILR4 = 0xFF;    // IRQ16: I2C ch0
                   // IRQ17: none
                   // IRQ18: 10-bit A/D-converter
                   // IRQ19: Timebase timer

    ILR5 = 0xFF;    // IRQ20: Watch timer
                   // IRQ21: none
                   // IRQ22: none
                   // IRQ23: Flash Memory
}

/*-----
   Prototypes

   Add your own prototypes here. Each vector definition needs is proto-
   type. Either do it here or include a header file containing them.
   -----*/
__interrupt void DefaultIRQHandler(void);

/*-----
   Vector definiton

   Use following statements to define vectors.
   All resource related vectors are predefined.
   Remaining software interrupts can be added hereas well.
   -----*/
#pragma intvect DefaultIRQHandler 0 // IRQ0:  external interrupt ch0 | ch4
#pragma intvect DefaultIRQHandler 1 // IRQ1:  external interrupt ch1 | ch5

```

```
#pragma intvect DefaultIRQHandler 2 // IRQ2: external interrupt ch2 | ch6
#pragma intvect DefaultIRQHandler 3 // IRQ3: external interrupt ch3 | ch7

#pragma intvect DefaultIRQHandler 4 // IRQ4: UART/SIO ch0
#pragma intvect DefaultIRQHandler 5 // IRQ5: 8/16-bit timer ch0 (lower)
#pragma intvect DefaultIRQHandler 6 // IRQ6: 8/16-bit timer ch0 (upper)
#pragma intvect DefaultIRQHandler 7 // IRQ7: Output Compare ch0

#pragma intvect DefaultIRQHandler 8 // IRQ8: Output Compare ch1
#pragma intvect DefaultIRQHandler 9 // IRQ9: none
#pragma intvect DefaultIRQHandler 10 // IRQ10: Voltage Compare ch0
#pragma intvect DefaultIRQHandler 11 // IRQ11: Voltage Compare ch1

#pragma intvect DefaultIRQHandler 12 // IRQ12: Voltage Compare ch2
#pragma intvect DefaultIRQHandler 13 // IRQ13: Voltage Compare ch3
#pragma intvect DefaultIRQHandler 14 // IRQ14: 16-bit free run timer
#pragma intvect DefaultIRQHandler 15 // IRQ15: 16-bit PPG0

#pragma intvect DefaultIRQHandler 16 // IRQ16: I2C ch0
#pragma intvect DefaultIRQHandler 17 // IRQ17: none
#pragma intvect DefaultIRQHandler 18 // IRQ18: 10-bit A/D-converter
#pragma intvect DefaultIRQHandler 19 // IRQ19: Timebase timer

#pragma intvect DefaultIRQHandler 20 // IRQ20: Watch timer
#pragma intvect DefaultIRQHandler 21 // IRQ21: none
#pragma intvect DefaultIRQHandler 22 // IRQ22: none
#pragma intvect DefaultIRQHandler 23 // IRQ23: Flash Memory
```

```
/*-----
    DefaultIRQHandler()
```

This function is a placeholder for all vector definitions.
 Either use your own placeholder or add necessary code here
 (the real used resource interrupt handlers should be defined in the main.c).

```
-----*/
```

```
__interrupt void DefaultIRQHandler(void)
{
    __DI();                // disable interrupts
    while(1)
        __wait_nop();      // halt system
}
```